Hspec language definition[1]
– version 1.1.0 –

Mihai Codescu[1] and Răzvan Diaconescu[2]

[1]Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Research Group of the project PED-0494, Bucharest, Romania
[2]Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Bucharest, Romania

[1]This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI - UEFISCDI, project number PN-III-P2-2.1-PED-2016-0494, within PNCDI III.

Chapter 1

Hspec overview

A Hspec document consists of either specification of new hybrid logics or specification of reconfigurable systems in a hybrid logic.

Hybrid logic We introduce a declarative syntax for specifying the parameters of the generic hybridization method. They are:

- the name of the new hybridized logic,
- the name of the logic being hybridzed,
- the kinds of symbols allowed to appear in a quantification,
- the constraints made on the models of the logic.

We make the assumption that a library of possible constraints for each base logic is available, and the user must choose among the constraints of the specified base logic when making a new hybridization. Two types of constraints are possible:

- on the accessibility relations [3]:
  - reflexive: $(\forall w)R(w, w)$
  - symmetric: $(\forall w_1, w_2)R(w_1, w_2) \implies R(w_2, w_1)$
  - transitive: $(\forall w_1, w_2, w_3)R(w_1, w_2) \wedge R(w_2, w_3) \implies R(w_1, w_3)$
  - serial: $(\forall w_1)(\exists w_2)R(w_1, w_2)$
  - Euclidean: $(\forall w_1, w_2, w_3)R(w_1, w_2) \wedge R(w_1, w_3) \implies R(w_2, w_3)$
  - functional: $(\forall w_1)(\exists! w_2)R(w_1, w_2)$
  - linear: $(\forall w_1, w_2, w_3)(R(w_1, w_2) \wedge R(w_1, w_3)) \implies (R(w_2, w_3) \vee R(w_3, w_2) \vee @_{w_2} w_3)$
  - total: $(\forall w_1, w_2)R(w_1, w_2) \vee R(w_2, w_1)$

  where w, w_1, w_2, w_3 are worlds and R is the accessibility relation on worlds.

- on the local models:
  - the set of worlds of each local model is the same
  - the nominals are interpreted in the same way in each local model
  - symbols of some kind are interpreted in the same way in each local model
  - partial functions are defined on the same elements in each local model

Alternatively, one can add further semantic constraints or other kinds of symbols used in quantifications on an existing hybridized logic.

Hspec specifications Hspec basic specifications over a hybrid logic have three parts:

- the name of the hybrid logic,
- the name of a specification in the base logic of the hybridized logic, containing the data part of the specification,
- a configuration part, consisting of declarations of state names and events and sentences in the hybrid logic.

For structuring, we will make use of the DOL language [5]. DOL is a metalanguage for structuring of ontologies, specifications and MDE models, independent of the formalism used at the basic level. A DOL structured specification can contain parts written in different logics. In our setting, we will only make use of homogeneous structuring, where all specifications appearing in a structured specifications are in the same logic.

Chapter 2

Hspec syntax

2.1 Abstract syntax

2.1.1 Hspec documents

```
Document         ::= HLogicDef | HDef*

HLogicDef        ::= hlogic LogicName HLogic

HLogic           ::= hybridizeBase
                          LogicName
                          [QuantRestr*]
                          [SemConstr*]
                   | addQuantOrConstr
                          LogicName
                          [QuantRestr*]
                          [SemConstr*]

LogicName        ::= Name
QuantRestr       ::= Name

SemConstr        ::= Reflexive | Transitive | Symmetric
                   | Serial | Euclidean | Functional
                   | Linear | Total
                   | SameInterpretation Kind
                   | SameDomain PartialOrRigid

PartialOrRigid   ::= partial | rigid partial
Kind             ::= world | nominal
                   | Name | rigid Name
```

2.1.2 Hspec structured specifications

```
HDef           ::= hDef SpecName HSpec

HSpec          ::= BaseSpec
                 | HBasicSpec
                 | extension HSpec HBasicSpec
                 | union HSpec HSpec
                 | renaming HSpec SymbolMap

BaseSpec       ::= <logic specific syntax>

SymbolMap      ::= symbolMap (Id, Id)+
```

2.1.3 Hspec basic specifications

```
HBasicSpec     ::= hBasicSpec LogicPart DataPart ConfigPart
SpecName       ::= Name
LogicPart      ::= logic Name
DataPart       ::= data SpecName
ConfigPart     ::= configuration HybridDecl* HSen*

HybridDecl     ::= NomDecl | ModDecl
NomDecl        ::= nominals Name+
ModDecl        ::= modalities (Name, Nat)+

HSen           ::= BasicSen | Nominal | Negation
                 | Conjunction | Disjunction | Implication
                 | AtSen | BoxSen| DiamondSen
                 | QuantifiedSen

BasicSen       ::= <logic specific syntax>
Nominal        ::= Id
Negation       ::= negation HSen
Conjunction    ::= conjunction HSen HSen
Disjunction    ::= disjunction HSen HSen
Implication    ::= implication HSen HSen
AtSen          ::= at Id HSen
BoxSen         ::= box Id HSen+
DiamondSen     ::= diamond Id HSen+
QuantifiedSen  ::= quantification QualQuant QualNom HSen
                 | quantification QualQuant BaseSpec HSen

QualQuant      ::= qualQuant Quant LogicName
Quant          ::= forallH | existsH
```

```
QualNom        ::= nominals LogicName Name+

Id             ::= QualName | Name
Name           ::= %%letters , digits and special characters
QualName       ::= qualName Name LogicName
```

2.2 Relation with DOL

Hspec can be regarded as an extension of a fragment of DOL. This can be explained as follows:

- At the level of libraries, we add a new type of library item for logic definitions. `HDef` is a renaming of `OMSDefinition`.
- At the level of structured specifications, all Hspec constructs are inherited from DOL.
- At the level of basic specifications, a Hspec specification

  ```
  spec S = logic: L data: D configuration: C
  ```

  can be equivalently written in DOL as

  ```
  spec S = logic L : { data D C }
  ```

  .

2.3 Concrete syntax

2.3.1 Hspec documents

```
Document        ::= HLogicDef | HDef*

HLogicDef       ::= 'hlogic' LogicName '=' HLogic

HLogic          ::= 'base:' LogicName '.'
                    ['quant:' QuantRestr* '.']
                    ['constr:' SemConstr* '.']
                  | 'hlogic:' LogicName '.'
                    ['quant:' QuantRestr* '.']
                    ['constr:' SemConstr* '.']

LogicName       ::= Name
QuantRestr      ::= Name
SemConstr       ::= 'Reflexive' | 'Transitive' | 'Symmetric'
                  | 'Serial' | 'Euclidean' | 'Functional'
                  | 'Linear' | 'Total'
                  | 'SameInterpretation(' Kind+ ')'
                  | 'SameDomain(' PartialOrRigid ')'

PartialOrRigid  ::= 'partial' | 'rigid partial'
Kind            ::= 'world' | 'nominal'
                  | Name | 'rigid' Name
```

2.3.2 Hspec structured specifications

```
HSpec           ::= BaseSpec
                  | HBasicSpec
                  | HSpec 'then' HBasicSpec
                  | HSpec 'and' HSpec
                  | HSpec 'with' SymbolMap

BaseSpec        ::= <logic specific syntax>

SymbolMap       ::= (Id '|->' Id)+
```

2.3.3 Hspec basic specifications

Developer-oriented notation

```
HBasicSpec      ::= LogicPart DataPart ConfigPart
SpecName        ::= Name
LogicPart       ::= 'logic:' Name
DataPart        ::= 'data:' SpecName
```

```
ConfigPart      ::= 'configuration:' HybridDecl* HSen*

HybridDecl      ::= NomDecl | ModDecl
NomDecl         ::= 'states' Name, ..., Name
ModDecl         ::= 'events' ModItem, ..., ModItem
ModItem         ::= Name ':' Nat

HSen            ::= BasicSen | Nominal | Negation
                  | Conjunction | Disjunction | Implication
                  | AtSen | BoxSen| DiamondSen
                  | QuantifiedSen

BasicSen        ::= <logic specific syntax>
Nominal         ::= Id
Negation        ::= 'not' HSen
Conjunction     ::= HSen '/\' HSen
Disjunction     ::= HSen '\/' HSen
Implication     ::= HSen '=>' HSen
AtSen           ::= 'At' Id ':' HSen
BoxSen          ::= 'Through' Id 'always' HSen+
DiamondSen      ::= 'Through' Id 'sometimes' HSen+
QuantifiedSen   ::= QualQuant QualNom HSen
                  | QualQuant BaseSpec HSen

QualQuant       ::= 'forallH.'LogicName | 'existsH.'LogicName

QualNom         ::= 'states.'LogicName Name+

Id              ::= QualName | Name
Name            ::= %%list of letters, digits and special characters
QualName        ::= Name '::' LogicName
```

Abbreviations:

At s
: sen_1
: sen_2
...
: sen_k
end

for

At s : sen_1
At s : sen_2
...
At s : sen_k
end

```
Through e, only sen
```

for

```
Through e, sometimes sen
/\
Through e, always sen
```

Mathematical-oriented notation

```
HBasicSpec     ::= LogicPart DataPart ConfigPart
SpecName       ::= Name
LogicPart      ::= `logic:` Name
DataPart       ::= `data:` SpecName
ConfigPart     ::= `configuration:` HybridDecl* HSen*

HybridDecl     ::= NomDecl | ModDecl
NomDecl        ::= `nominals` Name, ..., Name
ModDecl        ::= `modalities` ModItem, ..., ModItem
ModItem        ::= Name `:` Nat

HSen           ::= BasicSen | Nominal | Negation
                 | Conjunction | Disjunction | Implication
                 | AtSen | BoxSen| DiamondSen
                 | QuantifiedSen

BasicSen       ::= <logic specific syntax>
Nominal        ::= Id
Negation       ::= `not` HSen
Conjunction    ::= HSen `/\` HSen
Disjunction    ::= HSen `\/` HSen
Implication    ::= HSen `=>` HSen
AtSen          ::= `@` Id `:` HSen
BoxSen         ::= `[` Id `]` HSen+
DiamondSen     ::= `<` Id `>` HSen+
QuantifiedSen  ::= QualQuant QualNom HSen
                 | QualQuant BaseSpec HSen

QualQuant      ::= `forallH.`LogicName | `existsH.`LogicName

QualNom        ::= `nominals.`LogicName Name+

Id             ::= QualName | Name
Name           ::= %%list of letters, digits and special characters
QualName       ::= Name `::` LogicName
```

Chapter 3

Hspec semantics

3.1 Foundations

Definition 3.1.1. Let $\mathbb{S}et$ be the category[1] having all small sets as objects and functions as arrows, and let $\mathbb{C}at$ be the category of categories and functors.[2] An *institution* [2] is a tuple $I = (\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models)$ consisting of the following:

- a category Sign of *signatures* and *signature morphisms*,
- a functor $\mathsf{Sen}\colon \mathsf{Sign} \longrightarrow \mathbb{S}et$ giving, for each signature Σ, the set of *sentences* $\mathsf{Sen}(\Sigma)$, and for each signature morphism $\sigma : \Sigma \to \Sigma'$, the *sentence translation map* $\mathsf{Sen}(\sigma) : \mathsf{Sen}(\Sigma) \to \mathsf{Sen}(\Sigma')$, where often $\mathsf{Sen}(\sigma)(\varphi)$ is written as $\sigma(\varphi)$,
- a functor $\mathsf{Mod} : \mathsf{Sign}^{op} \to \mathbb{C}at$ giving, for each signature Σ, the category of *models* $\mathsf{Mod}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *reduct functor* $\mathsf{Mod}(\sigma) : \mathsf{Mod}(\Sigma') \to \mathsf{Mod}(\Sigma)$, where often $\mathsf{Mod}(\sigma)(M')$ is written as $M'|_\sigma$, and $M'|_\sigma$ is called the *σ-reduct* of M', while M' is called a *σ-expansion* of $M'|_\sigma$,
- a satisfaction relation $\models_\Sigma \subseteq |\mathsf{Mod}(\Sigma)| \times \mathsf{Sen}(\Sigma)$ for each $\Sigma \in |\mathsf{Sign}|$,

such that for each $\sigma\colon \Sigma \longrightarrow \Sigma'$ in Sign the following *satisfaction condition* holds:

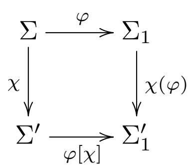
$$(\star) \qquad M' \models_{\Sigma'} \sigma(\varphi) \text{ iff } M'|_\sigma \models_\Sigma \varphi$$

for each $M' \in |\mathsf{Mod}(\Sigma')|$ and $\varphi \in \mathsf{Sen}(\Sigma)$, expressing that truth is invariant under change of notation and context. □

[1] See [1, 4] for an introduction into category theory.

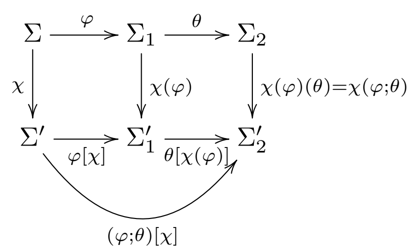
[2] Strictly speaking, $\mathbb{C}at$ is not a category but only a so-called quasicategory, which is a category that lives in a higher set-theoretic universe.

Definition 3.1.2. Let Sign be a category and let $\mathcal{D}$ be a subclass of arrows from Sign. $\mathcal{D}$ is called a *quantification space* if for any $\chi : \Sigma \to \Sigma' \in \mathcal{D}$ and any $\varphi : \Sigma \to \Sigma_1$, there is a designated pushout



with $\chi(\varphi) \in \mathcal{D}$ and such that

- the horizontal composition of designated pushouts is a designated pushout, i.e. in the following digram



  we have that $\chi(\varphi)(\theta) = \chi(\varphi;\theta)$ and $(\varphi;\theta)[\chi] = \varphi[\chi];\theta[\chi(\varphi)]$,

- $\chi(1_\Sigma) = \chi$ and $1_\Sigma[\chi] = 1_{\Sigma'}$

Definition 3.1.3. An institution with kinded symbols is a tuple $(\mathcal{I}, \mathbf{kind} : \mathbf{Symbols} \to \mathbf{Kinds}, \mathbf{Sym} : \mathsf{Sign} \to \mathbf{Symbols})$ where

- $\mathcal{I}$ is an institution,

- $\mathbf{kind} : \mathbf{Symbols} \to \mathbf{Kinds}$ is a function from a set **Symbols** of symbols to a set **Kinds** of kinds, giving the kind of each symbol,

- $\mathbf{Sym} : \mathsf{Sign} \to \mathbf{Symbols}$ is a faithful functor[3] assigning to each signature Σ a set $\mathbf{Sym}(\Sigma) \subseteq \mathbf{Symbols}$ of Σ-symbols and to each $\sigma : \Sigma \to \Sigma'$ a function $\mathbf{Sym}(\sigma) : \mathbf{Sym}(\Sigma) \to \mathbf{Sym}(\Sigma')$ such that for each symbol $s \in \mathbf{Sym}(\Sigma)$, $\mathbf{kind}(\mathbf{Sym}(\sigma)(s)) = \mathbf{kind}(s)$.

The semantics of H specifications is given in the context of a *heterogeneous logical environment* Γ, consisting of

- a list Γ_{log} of institutions with kinded symbols together with

[3] A functor is faithful if it is injective when restricted to each set of morphisms that have a given source and target.

  - a partial function *baseLogic* giving the base institution for a hybridised institution,
  - a function $sem^{\mathcal{I}}$ giving the semantics of a basic specification in $\mathcal{I}$ and a function $sem^{\mathcal{I}}_{\Sigma}$ giving the semantics of a basic specification in the context Σ of previous declarations.

- a mapping Γ_{def} from specification names to semantics of specifications, giving access to previous declarations.

We make the assumption that the category of signatures of each institution of the logical environment admits unions and differences.

If Γ is a heterogeneous logical environment, $\Gamma_{log}[L \mapsto \mathcal{I}]$ extends Γ with a new institution $\mathcal{I}$ named L, and $\Gamma_{def}[S \mapsto (\mathcal{I}, \Sigma, \mathcal{M})]$ extends Γ with a new specification named S whose semantics is $(\mathcal{I}, \Sigma, \mathcal{M})$. The lookup functions for institutions and specifications are denoted $\Gamma(L)$ and $\Gamma(S)$ respectively.

For each syntactic category in the abstract syntax, we specify a semantic domain, giving the possible values for the semantics. The semantics is defined using semantic rules, involving a function *sem*, whose last argument is always the syntactic entity for which semantics is defined, while the other arguments determine the context in which semantics is defined.

3.2 Hspec documents

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{HLogicDef}) & = \Gamma' \\ & : \mathit{LogicalEnv} \end{array}}$$

$$sem(\Gamma, \texttt{hlogic}\ L\ I) = \Gamma_{log}[L \mapsto \mathcal{I}]$$

where

- L is a name that is not in the domain of Γ,
- $sem(\Gamma, I) = \mathcal{I}$,

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{HLogic}) & = \mathcal{I} \\ & : \mathit{Institution} \end{array}}$$

$$sem(\Gamma, \texttt{hybridizeBase}\ L\ Q\ C) = \mathcal{I}'$$

where

- $\Gamma(L) = \mathcal{I} = (\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models)$,
- $sem(\mathcal{I}, Q) = \mathcal{D}$,
- $\mathcal{I}'$ is the institution defined by:

1. the category Sign' of $\mathcal{I}'$-signatures has as objects triples of the form $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$ where Σ is a signature in $\mathcal{I}$, Nom is a set (of state names, usually called *nominals*) such that Nom and $\mathsf{Sen}(\Sigma)$ are disjoint and $\Lambda = \{\Lambda_n\}_{n\in\mathbb{N}}$ is a $\mathbb{N}$-sorted set (of *modalities*). A signature morphism $\varphi : \Delta^1 \to \Delta^2$ betweeen two $\mathcal{I}'$-signatures $\Delta^1 = (\Sigma^1, \mathsf{Nom}^1, \Lambda^1)$ and $\Delta^2 = (\Sigma^2, \mathsf{Nom}^2, \Lambda^2)$ consists of a $\mathcal{I}$-signature morphism $\varphi^{\mathsf{Sign}} : \Sigma^1 \to \Sigma^2$, a function $\varphi^{\mathsf{Nom}} : \mathsf{Nom}^1 \to \mathsf{Nom}^2$ and a family of functions $\varphi^{\Lambda} = \{\varphi^{\Lambda}_n : \Lambda^1_n \to \Lambda^2_n\}_{n\in\mathbb{N}}$.
2. if $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$ is a $\mathcal{I}'$-signature, the set $\mathsf{Sen}'(\Delta)$ of Δ-sentences is the least set such that:
   - $i \in \mathsf{Sen}'(\Delta)$, for $i \in \mathsf{Nom}$,
   - $e \in \mathsf{Sen}'(\Delta)$, for $e \in \mathsf{Sen}(\Sigma)$,
   - $\neg\xi \in \mathsf{Sen}'(\Delta)$, for $\xi \in \mathsf{Sen}'(\Delta)$,
   - $\xi_1 \star \xi_2 \in \mathsf{Sen}'(\Delta)$, for $\xi_1, \xi_2 \in \mathsf{Sen}'(\Delta)$ and $\star \in \{\wedge, \vee, \implies\}$,
   - $@_i\xi \in \mathsf{Sen}'(\Delta)$, for $\xi \in \mathsf{Sen}'(\Delta)$ and $i \in \mathsf{Nom}$,
   - $[\lambda](\xi_1, \ldots, \xi_n)$ and $\langle\lambda\rangle(\xi_1, \ldots, \xi_n) \in Sen'(\Delta)$, for $\lambda \in \Lambda_{n+1}$ and $\xi_1, \ldots, \xi_n \in \mathsf{Sen}'(\Delta)$ and
   - $(\forall\chi)\xi', (\exists\chi)\xi' \in \mathsf{Sen}'(\Delta)$, for $\chi : \Delta \to \Delta' \in \mathcal{D}$ and $\xi' \in \mathsf{Sen}'(\Delta')$.

   If $\varphi : \Delta \to \Delta_1$ is an I'-signature morphism, the sentence translation function $\mathsf{Sen}'(\varphi) : \mathsf{Sen}'(\Delta) \to \mathsf{Sen}'(\Delta_1)$ is defined by
   - $\mathsf{Sen}'(\varphi)(i) = \varphi^{\mathsf{Nom}}(i)$, for $i \in \mathsf{Nom}$,
   - $\mathsf{Sen}'(\varphi)(e) = \mathsf{Sen}(\varphi^{\mathsf{Sign}})(e)$, for $e \in \mathsf{Sen}(\Sigma)$,
   - $\mathsf{Sen}'(\varphi)(\neg\xi) = \neg\mathsf{Sen}'(\varphi)(\xi)$, for $\xi \in \mathsf{Sen}'(\Delta)$,
   - $\mathsf{Sen}'(\varphi)(\xi_1 \star \xi_2) = \mathsf{Sen}'(\varphi)(\xi_1) \star \mathsf{Sen}'(\varphi)(\xi_2)$, for $\xi_1, \xi_2 \in \mathsf{Sen}'(\Delta)$ and $\star \in \{\wedge, \vee, \implies\}$,
   - $\mathsf{Sen}'(\varphi)(@_i\xi) = @_{\varphi^{\mathsf{Nom}}(i)}\mathsf{Sen}'(\varphi)(\xi)$, for $\xi \in \mathsf{Sen}'(\Delta)$ and $i \in \mathsf{Nom}$,
   - $\mathsf{Sen}'(\varphi)([\lambda](\xi_1, \ldots, \xi_n)) = [\varphi^{\Lambda}(\lambda)](\mathsf{Sen}'(\varphi)(\xi_1), \ldots, \mathsf{Sen}'(\varphi)(\xi_n))$ and $\mathsf{Sen}'(\varphi)(\langle\lambda\rangle(\xi_1, \ldots, \xi_n)) = \langle\varphi^{\Lambda}(\lambda)\rangle(\mathsf{Sen}'(\varphi)(\xi_1), \ldots, \mathsf{Sen}'(\varphi)(\xi_n))$ for $\lambda \in \Lambda_{n+1}$ and $\xi_1, \ldots, \xi_n \in \mathsf{Sen}'(\Delta)$ and
   - $\mathsf{Sen}'(\varphi)((\forall\chi)\xi') = (\forall\chi(\varphi))\mathsf{Sen}'(\varphi[\chi])(\xi')$ and $\mathsf{Sen}'(\varphi)((\exists\chi)\xi') = (\exists\chi(\varphi))\mathsf{Sen}'(\varphi[\chi])(\xi')$, for $\chi : \Delta \to \Delta' \in \mathcal{D}$ and $\xi' \in \mathsf{Sen}'(\Delta')$.
3. for each $\mathcal{I}'$-signature Δ, the category $\mathsf{Mod}'(\Delta)$ has as objects pairs (W, M) where $|W|$ is a set (of *states*), for each $i \in \mathsf{Nom}$, $W_i \in |W|$, for each $\lambda \in \Lambda_n$, W_λ is an n-ary relation on $|W|$ and $M_w \in \mathsf{Mod}(\Sigma)$ for each $w \in |W|$. A model homomorphism $h : (W, M) \to (W', M')$ consists of a function $h_{st} : |W| \to |W'|$ such that $h_{st}(W_i) = W'_i$ and $W_\lambda(x_1, \ldots, x_n) \implies W'_\lambda(h_{st}(x_1), \ldots, h_{st}(x_n))$ for $x_1, \ldots, x_n \in |W|$ and $\lambda \in \Lambda_n$ and a natural transformation $h : M \Rightarrow M' \circ h_{st}$, i.e. a family of $\mathcal{I}$-homomorphisms $h_w : M_w \to M'_{h_{st}(w)}$ for each $w \in |W|$.

   If $\varphi : \Delta \to \Delta'$ is a signature morphism and (W', M') is a Δ'-model, its φ-reduct $(W', M')|_\varphi = (W, M)$ is defined as follows
   - $|W| = |W'|$, $W_i = W_{\varphi^{nom}(i)}$ and $W_\lambda = W'_{\varphi^{\Lambda}(\lambda)}$

- for each $w \in |W|$, $M_w = M'_w|_{\varphi^{sig}}$.

The list of semantic constraints determines the following restriction on the classes of models:

Semantic constraint	Restriction on binary modalities
`Reflexive`	All binary modalities must be reflexive
`Transitive`	All binary modalities must be transitive
`Symmetric`	All binary modalities must be symmetric
`Serial`	All binary modalities must be serial
`Euclidean`	All binary modalities must be Euclidean
`Functional`	All binary modalities must be functional
`Linear`	All binary modalities must be linear
`Total`	All binary modalities must be total

Semantic constraint	Restriction on local models
`SameInterpretation world`	Same set of worlds
`SameInterpretation nominal`	Nominals have the same interpretation
`SameInterpretation` k	Symbols of kind k have the same interpretation
`SameDomain partial`	Partial functions are defined on same arguments.
`SameDomain rigid partial`	Rigid partial functions are defined on same arguments.

4. for a signature Δ, a Δ-model (W, M) and a world $w \in W$, we define the satisfaction of a sentence in the world w as follows:
   - $(W, M) \models^w i$ iff $W_i = w$, for $i \in \mathsf{Nom}$,
   - $(W, M) \models^w e$ iff $M_w \models e$, for $e \in \mathsf{Sen}(\Sigma)$,
   - $(W, M) \models^w \neg\xi$ iff $(W, M) \not\models^w \xi$, for $\xi \in \mathsf{Sen}'(\Delta)$,
   - $(W, M) \models^w \xi_1 \wedge \xi_2$ iff $(W, M) \models^w \xi_1$ and $(W, M) \models^w \xi_2$, for $\xi_1, \xi_2 \in \mathsf{Sen}'(\Delta)$,
   - $(W, M) \models^w \xi_1 \vee \xi_2$ iff $(W, M) \models^w \xi_1$ or $(W, M) \models^w \xi_2$, for $\xi_1, \xi_2 \in \mathsf{Sen}'(\Delta)$,
   - $(W, M) \models^w \xi_1 \implies \xi_2$ iff $(W, M) \models^w \xi_2$ whenever $(W, M) \models^w \xi_1$, for $\xi_1, \xi_2 \in \mathsf{Sen}'(\Delta)$,
   - $(W, M) \models^w @_i\xi$ iff $(W, M) \models^{W_i} \xi$, for $\xi \in \mathsf{Sen}'(\Delta)$ and $i \in \mathsf{Nom}$,
   - $(W, M) \models^w [\lambda](\xi_1, \ldots, \xi_n)$ iff for each $w_1, \ldots, w_n \in |W|$ such that $W_\lambda(w, w_1, \ldots, w_n)$ we have that $(W, M) \models^{w_i} \xi_i$ for some $i = 1, \ldots, n$,
   - $(W, M) \models^w \langle\lambda\rangle(\xi_1, \ldots, \xi_n)$ iff exists $w_1, \ldots, w_n \in |W|$ such that $W_\lambda(w, w_1, \ldots, w_n)$ and $(W, M) \models^{w_i} \xi_i$ for each $i = 1, \ldots, n$,
   - $(W, M) \models^w (\forall\chi)\xi'$ iff for each χ-expansion (W', M') of (W, M), we have that $(W', M') \models^w \xi'$, where $\chi : \Delta \to \Delta' \in \mathcal{D}$ and $\xi' \in \mathsf{Sen}'(\Delta')$
   - $(W, M) \models^w (\exists\chi)\xi'$ iff there is a χ-expansion (W', M') of (W, M) such that $(W', M') \models^w \xi'$, for $\chi : \Delta \to \Delta' \in \mathcal{D}$ and $\xi' \in \mathsf{Sen}'(\Delta')$.

   Then $(W, M) \models \xi$ iff $(W, M) \models^w \xi$ for any $w \in |W|$.

$$sem(\Gamma, \texttt{addQuantOrConstr}\ L\ Q\ C) = \mathcal{I}'$$

where

- $\Gamma(L) = \mathcal{I}$ and $baseLogic(L)$ is defined (which ensures that L is a hybridized institution),
- $\mathcal{I}'$ is the institutiton obtained by replacing the quantification space of $\mathcal{I}$ with its extension determined by $sem(\Gamma, Q)$,
- the classes of models of each signature are further restricted as given in C.

$$\boxed{\begin{array}{ll} sem(\mathcal{I}, \texttt{QuantRestr+}) & = \mathcal{D} \\ & : \mathit{MorphismsClass} \end{array}}$$

$$sem(\mathcal{I}, n_1\ n_2\ \ldots n_k) = \mathcal{D}$$

where $\mathcal{D}$ is the class of signature extensions in $\mathcal{I}$ with symbols whose kind is among $n_1, \ldots, n_k$. All kinds must be valid for $\mathcal{I}$, i.e. $n_i \in$ **Kinds** for each $i = 1, \ldots, k$.

3.3 Hspec structured specifications

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{HDef}) & = \Gamma' \\ & : \mathit{LogicalEnv} \end{array}}$$

$$sem(\Gamma, \texttt{hdef}\ N\ S) = \Gamma[N \mapsto (\mathcal{I}, \Sigma, \mathcal{M})]$$

where $sem(\Gamma, S) = (\mathcal{I}, \Sigma, \mathcal{M})$.

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{HSpec}) & = (\mathcal{I}, \Sigma, \mathcal{M}) \\ & : (\mathit{Institution}, \mathit{Signature}, \mathit{ModelClass}) \end{array}}$$

$$sem(\Gamma, baseSpec) = (\mathcal{I}, \Sigma, \mathcal{M})$$

where $\mathcal{I}$ is the logic of $baseSpec$ and $sem^{\mathcal{I}}(baseSpec) = (\Sigma, \mathcal{M})$.

$$sem(\Gamma, \texttt{extension}\ S1\ S2) = (\mathcal{I}, \Sigma, \mathcal{M})$$

where

- $sem(\Gamma, S1) = (\mathcal{I}, \Sigma_1, \mathcal{M}_1)$,
- $sem^{\mathcal{I}}_{\Sigma_1}(S2) = (\mathcal{I}, \Sigma, \mathcal{M}_2)$,
- $\mathcal{M} = \{M \in \mathcal{M}_2 \mid M|_{\Sigma_1} \in \mathcal{M}_1\}$

$$sem(\Gamma, \texttt{union}\ S1\ S2) = (\mathcal{I}, \Sigma, \mathcal{M})$$

where

- $sem(\Gamma, S1) = (\mathcal{I}, \Sigma_1, \mathcal{M}_1)$,
- $sem(\Gamma, S2) = (\mathcal{I}, \Sigma_2, \mathcal{M}_2)$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\mathcal{M} = \{M \in \mathsf{Mod}(\Sigma) \mid M|_{\Sigma_i} \in \mathcal{M}_i\}$

$$sem(\Gamma, \texttt{renaming}\ S1\ symmap) = (\mathcal{I}, \Sigma, \mathcal{M})$$

where

- $sem(\Gamma, S1) = (\mathcal{I}, \Sigma_1, \mathcal{M}_1)$,
- $sem(\Gamma, \Sigma_1, symmap) = \sigma : \Sigma_1 \to \Sigma$,
- $\mathcal{M} = \{M \in \mathsf{Mod}(\Sigma) \mid M|_{\Sigma_1} \in \mathcal{M}_1\}$

$$\boxed{\begin{array}{ll} sem(\Gamma, \Sigma, \texttt{SymbolMap}) & = \sigma : \Sigma \to \Sigma' \\ & : Morphism \end{array}}$$

3.4 Hspec basic specifications

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{HBasicSpec}) & = (\mathcal{I}, \Sigma, \mathcal{M}) \\ & : (Institution, Signature, ModelClass) \end{array}}$$

$$sem(\Gamma, \texttt{hBasicSpec}\ logicPart\ dataPart\ configPart) = (\mathcal{I}, \Sigma, \mathcal{M})$$

where

- $sem(\Gamma, logicPart) = \mathcal{I}$,
- $sem(\Gamma, \mathcal{I}, dataPart) = (\Sigma_{data}, \mathcal{M}_{data})$,
- $sem(\Gamma, \mathcal{I}, \Sigma_{data}, configPart) = (\Delta, \mathcal{M}')$
- $\mathcal{M} = \{M \in \mathcal{M}' \mid M|_{\Sigma_{data}} \in \mathcal{M}_{data}\}$

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{LogicPart}) & = \mathcal{I} \\ & : Institution \end{array}}$$

$$sem(\Gamma, \texttt{logic}\ L) = \Gamma(L)$$

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \mathcal{I}, \texttt{DataPart}) & = (\Sigma, \mathcal{M}) \\ & : (\mathit{Signature}, \mathit{ModelClass}) \end{array}}$$

$$\mathit{sem}(\Gamma, \mathcal{I}, \texttt{data}\ S') = (\Sigma, \mathcal{M})$$

where

- $\Gamma(S') = (\mathcal{I}', \Sigma, \mathcal{M})$,
- $\mathit{baseLogic}(\mathcal{I}) = \mathcal{I}'$.

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \mathcal{I}, \Sigma_{data}, \texttt{ConfigPart}) & = (\Delta, \mathcal{M}) \\ & : (\mathit{Signature}, \mathit{ModelClass}) \end{array}}$$

$$\mathit{sem}(\Gamma, \mathcal{I}, \Sigma_{data}, \texttt{configuration}\ \mathit{hdecls}\ \mathit{hsens}) = (\Delta, \mathcal{M})$$

where

- $\Delta_0 = (\Sigma_{data}, \emptyset, \{\emptyset\}_{n \in \mathbf{N}})$,
- $\mathit{sem}(\Gamma, \mathcal{I}, \Delta_0, \mathit{hdecls}) = \Delta$
- $\mathit{sem}(\Gamma, \mathcal{I}, \Delta, \mathit{hsens}) = \mathit{Ax}$
- $\mathcal{M} = \{M \in \mathsf{Mod}(\Delta) \mid M \models \mathit{Ax}\}$

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \mathcal{I}, \Delta_{init}, \texttt{HybridDecl+}) & = \Delta \\ & : \mathit{Signature} \end{array}}$$

$$\mathit{sem}(\Gamma, \mathcal{I}, \Delta_{init}, \mathit{hdecl}_1, \ldots, \mathit{hdecl}_n) = \Delta$$

where

- $\mathit{sem}(\Gamma, \mathcal{I}, \Delta_{init}, \mathit{hdecl}_1) = \Delta_1$
- $\ldots$
- $\mathit{sem}(\Gamma, \mathcal{I}, \Delta_{n-1}, \mathit{hdecl}_n) = \Delta$

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \mathcal{I}, \Delta_{init}, \texttt{HybridDecl}) & = \Delta \\ & : \mathit{Signature} \end{array}}$$

$$\mathit{sem}(\Gamma, \mathcal{I}, \Delta_{init}, \texttt{nominals}\ \mathit{id}_1 \ldots \mathit{id}_k) = \Delta$$

where

- $\Delta_{init} = (\Sigma, \mathsf{Nom}, \Lambda)$,
- id_i does not appear in Nom for $i = 1, \ldots, k$,

- $\Delta = (\Sigma, \mathsf{Nom} \cup \{id_i \mid i = 1, \ldots, k\}, \Lambda)$.

$$sem(\Gamma, \mathcal{I}, \Delta_{init}, \texttt{modalities}\ (id_1, n_1) \ldots (id_k, n_k)) = \Delta$$

where

- $\Delta_{init} = (\Sigma, \mathsf{Nom}, \Lambda)$,
- id_i does not appear in Λ for $i = 1, \ldots, k$,
- for $n \in \mathbb{N}$, $\Lambda'_n = \Lambda_n \cup \{id_j \mid (id_j, n)$ a newly declared modality$\}$,
- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda')$.

$$\boxed{\begin{array}{ll} sem(\Gamma, \mathcal{I}, \Delta, \texttt{HSen+}) & = Ax \\ & : Set(Sentence) \end{array}}$$

$$sem(\Gamma, \mathcal{I}, \Delta, hsen_1 \ldots hsen_n) = Ax$$

where

- $sem(\Gamma, \mathcal{I}, \Delta, hsen_1) = \xi_1$,
- $sem(\Gamma, \mathcal{I}, \Delta, hsen_2) = \xi_2$,
- $\ldots$
- $sem(\Gamma, \mathcal{I}, \Delta, hsen_n) = \xi_n$,
- $Ax = \{\xi_1, \ldots \xi_n\}$.

$$\boxed{\begin{array}{ll} sem(\Gamma, \mathcal{I}, \Delta, \texttt{HSen}) & = \xi \\ & : Sentence \end{array}}$$

$$sem(\Gamma, \mathcal{I}, \Delta, basicSen) = \xi$$

where

- $\Delta = (\Sigma, Nom, \Lambda)$,
- $baseLogic(\mathcal{I}) = \mathcal{I}'$,
- $sem(\Gamma, \mathcal{I}', \Sigma, basicSen) = \xi$.

$$sem(\Gamma, \mathcal{I}, \Delta, qid) = id.L$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- $sem(\Gamma, \mathcal{I}, \Delta, nominal, qid) = id.L$,

- if $\Gamma(L) = \mathcal{I}$, id must be in Nom, otherwise $id.L = sem(\Gamma, baseLogic(\mathcal{I}), \Sigma, qid)$

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{negation}\ sen) = \neg\xi$$

where $sem(\Gamma, \mathcal{I}, \Delta, sen) = \xi$.

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{conjunction}\ sen_1\ sen_2) = \xi_1 \wedge \xi_2$$

where $sem(\Gamma, \mathcal{I}, \Delta, sen_i) = \xi_i$, for $i = 1, 2$.

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{disjunction}\ sen_1\ sen_2) = \xi_1 \vee \xi_2$$

where $sem(\Gamma, \mathcal{I}, \Delta, sen_i) = \xi_i$, for $i = 1, 2$.

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{implication}\ sen_1\ sen_2) = \xi_1 \implies \xi_2$$

where $sem(\Gamma, \mathcal{I}, \Delta, sen_i) = \xi_i$, for $i = 1, 2$.

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{at}\ qid\ sen) = @_{id}\xi$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- $sem(\Gamma, \mathcal{I}, \Delta, nominal, qid) = id.L$,
- if $\Gamma(L) = \mathcal{I}$ then if id is in Nom, $sem(\Gamma, \mathcal{I}, \Delta, sen) = \xi$
- if $\Gamma(L) \neq \mathcal{I}$, then $@_{id}\xi = sem(\Gamma, baseLogic(\mathcal{I}), \Sigma, \texttt{at}\ qid\ sen)$.

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{box}\ qid\ sen_1 \ldots sen_n) = [id](\xi_1, \ldots \xi_n)$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- $sem(\Gamma, \mathcal{I}, \Delta, modality, qid) = id.L$
- if $\Gamma(L) = \mathcal{I}$ then if id is in Λ_{n+1}, $sem(\Gamma, \mathcal{I}, \Delta, sen_i) = \xi_i$ for $i = 1, n$,
- if $\Gamma(L) \neq \mathcal{I}$, then $[id](\xi_1, \ldots \xi_n) = sem(\Gamma, baseLogic(\mathcal{I}), \sigma, \texttt{box}\ qid\ sen_1 \ldots sen_n)$

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{diamond}\ qid\ sen_1 \ldots sen_n) = \langle id \rangle(\xi_1, \ldots \xi_n)$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- $sem(\Gamma, \mathcal{I}, \Delta, modality, qid) = id.L$
- if $\Gamma(L) = \mathcal{I}$ then if id is in Λ_{n+1}, $sem(\Gamma, \mathcal{I}, \Delta, sen_i) = \xi_i$ for $i = 1, n$,

- if $\Gamma(L) \neq \mathcal{I}$, then $\langle id \rangle(\xi_1, \ldots \xi_n) = sem(\Gamma, baseLogic(\mathcal{I}), \sigma, \texttt{diamond}\ qid\ sen_1 \ldots sen_n)$

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{quantification}\ qquant\ qnom\ sen) = \xi$$

where

- $sem(\Gamma, \mathcal{I}, \Delta, qquant) = (q, \mathcal{I}', \Delta')$,
- $sem(\Gamma, \mathcal{I}', \Delta', qnom) = \varphi : \Delta' \to \Delta''$
- $sem(\Gamma, \mathcal{I}', \Delta'', sen) = \xi'$,
- $\xi = \begin{cases} (\exists\varphi)\xi' & q = existsH \\ (\forall\varphi)\xi' & q = forallH \end{cases}$

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{quantification}\ qquant\ bspec\ sen) = \xi$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- $sem(\Gamma, \mathcal{I}, \Delta, qquant) = (q, \mathcal{I}', \Delta')$,
- if $\mathcal{I}'' = baseLogic(\mathcal{I}')$, $sem_{\Sigma}^{\mathcal{I}''}(bspec) = (\mathcal{I}'', \Sigma', \mathcal{M})$ such that for each $s \in \mathbf{Sym}(\Sigma') \setminus \mathbf{Sym}(\Sigma)$, quantification on symbols of kind $\mathbf{kind}(s)$ is legal in $\mathcal{I}'$ and $\mathcal{M} = \mathsf{Mod}(\Sigma')$[4]
- $\varphi : \Delta' \to \Delta''$ is the extension of Δ' with all symbols in $\mathbf{Sym}(\Sigma') \setminus \mathbf{Sym}(\Sigma)$.
- $sem(\Gamma, \mathcal{I}', \Delta'', sen) = \xi'$,
- $\xi = \begin{cases} (\exists\varphi)\xi' & q = existsH \\ (\forall\varphi)\xi' & q = forallH \end{cases}$

$$\boxed{\begin{array}{ll} sem(\Gamma, \mathcal{I}, \Delta, \texttt{QualQuant}) & = (q, \mathcal{I}', \Delta') \\ & : (Quantifier, Institution, Signature) \end{array}}$$

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{qualQuant}\ q\ L) = (quantH, \mathcal{I}', \Delta')$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- $\mathcal{I}' = \Gamma(L)$,
- $sem(q) = quantH$
- if $\mathcal{I}' = \mathcal{I}$, $\Delta' = \Delta$, otherwise $sem(\Gamma, \mathcal{I}, \Delta, \texttt{qualQuant}\ q\ L) = sem(\Gamma, \mathcal{I}', \Sigma, \texttt{qualQuant}\ q\ L)$

[4] This ensures that there are no axioms in *bspec*.

$$\boxed{\begin{array}{ll} sem(\texttt{Quant}) & = \mathit{forallH} \mid \mathit{existsH} \\ & : \mathit{Quantifier} \end{array}}$$

$$sem(\texttt{forallH}) = \mathit{forallH}$$

$$sem(\texttt{existsH}) = \mathit{existsH}$$

$$\boxed{\begin{array}{ll} sem(\Gamma, \mathcal{I}, \Delta, \texttt{QualNom}) & = \sigma \\ & : \mathit{Morphism} \end{array}}$$

$$sem(\Gamma, \mathcal{I}, \Delta, \texttt{nominals}\ L\ i_1\ \dots\ i_n) = \sigma$$

where

- $\Delta = (\Sigma, \mathsf{Nom}, \Lambda)$
- if $\Gamma(L) = \mathcal{I}$, $\sigma : \Delta \to (\Sigma, \mathsf{Nom} \cup \{i_1, \dots, i_n\}, \Lambda)$ is the extension of Δ with the nominal variables $i_1, \dots, i_n$,
- otherwise, let $\sigma' = sem(\Gamma, \mathit{baseLogic}(\mathcal{I}), \Sigma, \texttt{nominals}\ L\ i_1\ \dots\ i_n)$ and expand σ' to $\sigma : \Delta \to \Delta'$ by letting σ be the identity on nominals and modalities on a level of hybridization higher than the one given by L.

$$\boxed{\begin{array}{ll} sem(\Gamma, \mathcal{I}, \Delta, k, \texttt{Id}) & = \mathit{symName.qualification} \\ & : \mathit{Name.LogicName} \end{array}}$$

$$sem(\Gamma, \mathcal{I}, \Delta, k, \texttt{qualName}\ n1\ n2) = n1.n2$$

if $n1 \in \mathbf{Sym}(\Delta)$ and $\mathit{kind}(n1) = k$.

$$sem(\Gamma, \mathcal{I}, \Delta, k, n) = n.L$$

where if $\Delta = (\Sigma, Nom, \Lambda)$ if $n_1, \dots, n_k$ is the list of all symbols in $\mathbf{Symbols}(\Delta)$ with name n and kind k, then

- if $k = 1$, then L is the unique logic name such that $\Gamma(L) = \mathcal{I}$,
- if $k > 1$, if there exists a symbol n_i such that n_i is not a symbol in Σ, then L is the unique logic name such that $\Gamma(L) = \mathcal{I}$, otherwise $n.L = sem(\Gamma, \mathit{baseLogic}(\mathcal{I}), \Sigma, n)$.

Bibliography

[1] J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. Wiley, New York, 1990.

[2] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39:95–146, 1992.

[3] Michael Huth and Mark Dermot Ryan. *Logic in computer science - modelling and reasoning about systems (2. ed.)*. Cambridge University Press, 2004.

[4] S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

[5] T. Mossakowski, M. Codescu, F. Neuhaus, and O. Kutz. *The Road to Universal Logic–Festschrift for 50th birthday of Jean-Yves Beziau, Volume II*, chapter The distributed ontology, modelling and specification language - DOL. Studies in Universal Logic. Birkhäuser, 2015.